# Crystallization Learning with the Delaunay Triangulation

Gu Jiaqi

Co-work with Prof. Guosheng Yin

Department of Statistics and Actuarial Science, The University of Hong Kong

# Background

# Motivation

- Estimating the conditional expectation function $\mu(\cdot) = E(Y|\cdot)$ under a regression model,

$$y_i = \mu(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, n, \quad (n > d), \tag{1}$$

  where $\mathbf{x}_i$ is a $d$-dimensional feature point in $\mathscr{R}^d$, $y_i \in \mathscr{R}$ is the observed response, $\epsilon_1, \ldots, \epsilon_n \in \mathscr{R}$ are i.i.d. random errors with $E(\epsilon_i) = 0$ and $E(\epsilon_i^2) < \infty$.
- Existing nonparametric methods: $k$-nearest neighbor, kernel regression, local linear regression, regression tree, random forest.
  - Advantages: Model-free; Robust in interpolation.
  - Disadvantage: Sensitive to the data density of $\mathbf{x}_i$s

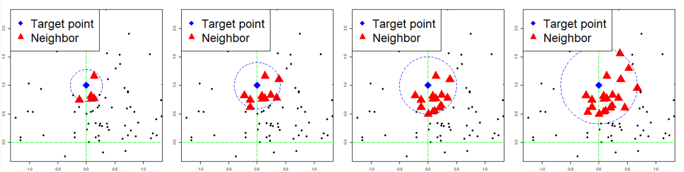
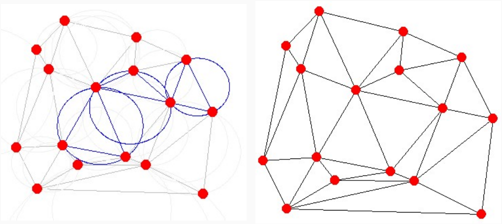



**Figure 1:** Neighbor data points of the target point $\mathbf{z}$ computed by the $k$-NN regression with $k = 5, 10, 15, 20$.

# Delaunay Interpolation: Delaunay Triangulation

- Let $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathscr{R}^d$. A triangulation of $\mathbb{X}$ is a mesh of disjoint $d$-simplices $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ which fully cover the convex hull of $\mathbb{X}$, $\mathcal{H}(\mathbb{X})$.
- Among all triangulations, the Delaunay triangulation is widely used for multivariate interpolation (de Berg et al., 2008) due to its smoothness.
- Let $\mathcal{B}_j$ be the open ball whose boundary is the circumscribed sphere of $\mathcal{S}_j$. The Delaunay triangulation of $\mathbb{X}$, denoted as $\mathcal{DT}(\mathbb{X})$, is any triangulation of $\mathbb{X}$ such that $\mathcal{B}_j \cap \mathbb{X} = \emptyset$ for $j = 1, \ldots, m$. (Empty-ball property)



(a) (b)

**Figure 2:** (a) Graphical illustration of the empty-ball property of the Delaunay triangulation; (b) the Delaunay triangulation.

# Delaunay Interpolation: Estimation

- Consider the data $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$ from model (1), the Delaunay interpolation estimates the conditional expectation $\mu(\mathbf{z})$ for all $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ in three steps:
  1. Construct the Delaunay triangulation $\mathcal{DT}(\mathbb{X})$;
  2. Find the simplex $\mathcal{S}(\mathbf{z}) \in \mathcal{DT}(\mathbb{X})$ such that $\mathbf{z} \in \mathcal{S}(\mathbf{z})$;
  3. Obtain the estimator $\hat{\mu}(\mathbf{z})$.
- Let $i_1(\mathbf{z}), \ldots, i_{d+1}(\mathbf{z})$ denote the indices corresponding to the data points of $\mathcal{S}(\mathbf{z})$. With $\gamma_1, \ldots, \gamma_{d+1} \in [0, 1]$ such that $\sum_{k=1}^{d+1} \gamma_k \mathbf{x}_{i_k(\mathbf{z})} = \mathbf{z}$ and $\sum_{k=1}^{d+1} \gamma_k = 1$, the estimator of de Berg et al. (2008) is

$$\hat{\mu}(\mathbf{z}) = \sum_{k=1}^{d+1} \gamma_k y_{i_k(\mathbf{z})}.$$

- However, the above approach requires a complete construction of $\mathcal{DT}(\mathbb{X})$, whose size grows exponentially with the dimension $d$. As a result, no existing algorithm is feasible when $d > 7$ due to the limitations of computation time/power and memory space.

# Delaunay Interpolation: DELAUNAYSPARSE Algorithm

- Recently, Chang et al. (2020) developed the DELAUNAYSPARSE algorithm to find $\mathcal{S}(\mathbf{z})$ for all $\mathbf{z} \in \mathcal{H}(\mathbb{X})$.
  1. Obtaining a seed Delaunay simplex $\mathcal{S}_{\text{seed}}$ close to $\mathbf{z}$;
  2. Growing neighbor Delaunay simplices of the explored ones in the direction of $\mathbf{z}$ recursively;
  3. Using the breadth first search to find $\mathcal{S}(\mathbf{z})$.

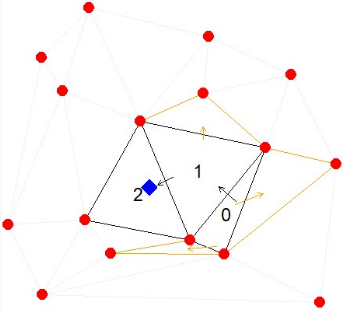



**Figure 3:** Graphical illustration of the DELAUNAYSPARSE algorithm.

# Methodology

# Crystallization Search for Delaunay Simplices

- Inspired by the DELAUNAYSPARSE algorithm, we develop the crystallization search to construct all the Delaunay simplices within the topological distance $L$ to $\mathcal{S}(\mathbf{z})$, denoted as $\mathcal{N}_L(\mathbf{z})$. (Computational Complexity: $\mathcal{O}(d^L n)$)

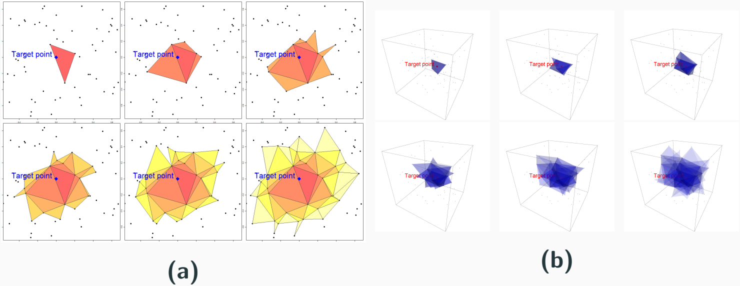



**Figure 4:** Crystallization search of $\mathcal{N}_L(\mathbf{z})$ with respect to a target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ and $L = 0, 1, 2$ (top row), $L = 3, 4, 5$ (bottom row) in $\mathscr{R}^2$ (a) and $\mathscr{R}^3$ (b).

# Crystallization Learning

- Let $\mathbb{V}_{\mathbf{z},L} = \cup_{\mathcal{S} \in \mathcal{N}_L(\mathbf{z})} \mathbb{V}(\mathcal{S})$ denote the set of all the data points of the simplices in $\mathcal{N}_L(\mathbf{z})$. We propose the crystallization learning to estimate $\mu(\mathbf{z})$ by fitting a local linear model, $\mu(\mathbf{z}) = \alpha + \boldsymbol{\beta}^\mathsf{T}\mathbf{z}$, to all the data points in $\mathbb{V}_{\mathbf{z},L}$ instead of only the $d+1$ data points of $\mathcal{S}(\mathbf{z})$.
- We estimate $\alpha$ and $\boldsymbol{\beta}$ via the weighted least squares approach,

$$(\hat{\alpha}, \hat{\boldsymbol{\beta}}) = \arg\min \sum_{\mathbf{x}_i \in \mathbb{V}_{\mathbf{z},L}} \mathrm{w}_{\mathbf{z},L}(\mathbf{x}_i)(y_i - \alpha - \boldsymbol{\beta}^\mathsf{T}\mathbf{x}_i)^2,$$

  where $\mathrm{w}_{\mathbf{z},L}(\mathbf{x}_i)$ is larger if $\mathbf{x}_i$ is closer to the target point $\mathbf{z}$ and shared by more simplices of $\mathcal{N}_L(\mathbf{z})$.
- As a small $L$ leads to overfitting and a large $L$ makes $\hat{\mu}(\cdot)$ overly smooth, we propose adapting the leave-one-out cross validation (LOO-CV) to select $L$ with respect to the target point $\mathbf{z}$

# Connection with Other Nonparametric Regression Methods

- Similar to existing nonparametric regression methods, our crystallization learning consists of three steps in estimating the conditional expectation $\mu(\mathbf{z})$:
  1. Selecting data points from $\mathbb{X}$ as the neighbors of $\mathbf{z}$ according to a specific criterion;
  2. Assigning weights to the selected neighbor data points;
  3. Fitting a local model to the selected neighbor data points.
- Since our crystallization learning and the existing methods mainly differ in the first two steps, we compare our crystallization learning with the $k$-nearest neighbor ($k$-NN) regression and the local linear regression in the computation of neighbor data points. We use the Euclidean distance in the $k$-NN regression and the Gaussian kernel in the local linear regression.

# Connection with Other Nonparametric Regression Methods

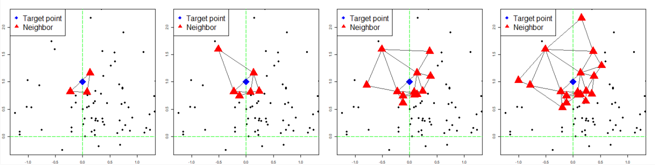
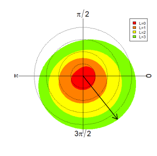
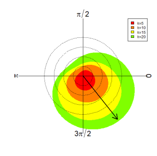


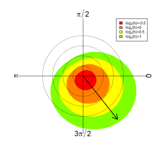

**Figure 5:** Neighbor data points of the target point $\mathbf{z}$ computed by the crystallization learning with $L = 0, 1, 2, 3$.




**(a)** Crystallization




**(b)** $k$-NN




**(c)** Local linear

**Figure 6:** Kernel density estimates of distributions of the directions from the target point $\mathbf{z}$ to its neighbor data points using different methods with different hyperparameter values. The arrow indicates the direction from the target point $\mathbf{z}$ to the sample mean of $\mathbb{X}$.

# Connection with Other Nonparametric Regression Methods

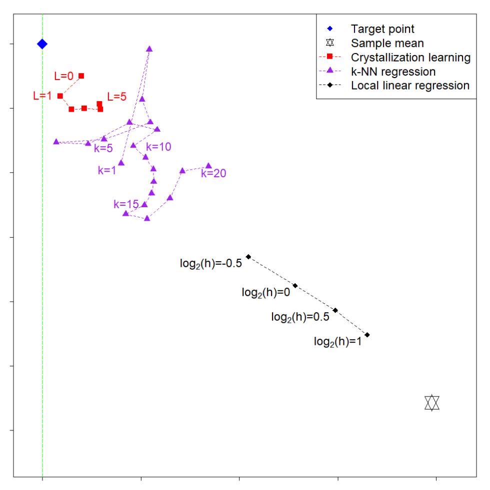



**Figure 7:** Paths of the (weighted) means of neighbor data points by different methods as the value of the hyperparameter increases.

# Experiments

# Experiments

- We conduct experiments on synthetic data under two different scenarios (general internal points of $\mathcal{H}(\mathbb{X})$, or jump points of the feature data density):
    1. to illustrate the effectiveness of the crystallization learning in estimating the conditional expectation function $\mu(\cdot)$;
    2. to evaluate the estimation accuracy of our approach in comparison with existing nonparametric regression methods, including the $k$-NN regression using the Euclidean distance, local linear regression using Gaussian kernel, multivariate kernel regression using Gaussian kernel (Hein, 2009) and Gaussian process models.
- We use the mean squared error (MSE) under the method $\mathcal{M}$,

$$\text{MSE}_{\mathcal{M}} = \frac{1}{100} \sum_{k=1}^{100} \{\hat{\mu}_{\mathcal{M}}(\mathbf{z}_k) - \mu(\mathbf{z}_k)\}^2,$$

  to evaluate the accuracy of the estimator $\hat{\mu}_{\mathcal{M}}(\cdot)$ at the target points $\mathbf{z}_1, \ldots, \mathbf{z}_{100} \in \mathcal{H}(\mathbb{X})$..
- We also apply our method to real data to investigate its empirical performance.

# Experiments on Synthetic Data

*Table 2.* Averaged values of $\log(\text{MSE})$ and standard deviations in parentheses using crystallization learning (CL) in comparison with $k$-NN ($k = 5, 10, k^*$, where $k^*$ equals the size of $\mathbb{V}_{\mathbf{z},L}$), local linear (LL) regression, kernel regression (KR) and Gaussian process (GP) in estimating $\mu(\cdot)$ under two scenarios, different sample sizes ($n$) and different dimensions of the feature space ($d$).

| $d$ | $n$ | $\log(\text{MSE}_{\text{CL}})$ | $\log\left(\frac{\text{MSE}_{5\text{-NN}}}{\text{MSE}_{\text{CL}}}\right)$ | $\log\left(\frac{\text{MSE}_{10\text{-NN}}}{\text{MSE}_{\text{CL}}}\right)$ | $\log\left(\frac{\text{MSE}_{k^*\text{-NN}}}{\text{MSE}_{\text{CL}}}\right)$ | $\log\left(\frac{\text{MSE}_{\text{LL}}}{\text{MSE}_{\text{CL}}}\right)$ | $\log\left(\frac{\text{MSE}_{\text{KR}}}{\text{MSE}_{\text{CL}}}\right)$ | $\log\left(\frac{\text{MSE}_{\text{GP}}}{\text{MSE}_{\text{CL}}}\right)$ |
|---|---|---|---|---|---|---|---|---|
| Scenario 1 (General internal points) | | | | | | | | |
| 5 | 200 | -1.11(0.21) | 0.23(0.09) | 0.12(0.09) | 0.33(0.11) | 0.56(0.11) | 0.57(0.11) | 0.24(0.18) |
| | 500 | -2.13(0.18) | 0.55(0.13) | 0.37(0.11) | 0.45(0.13) | 0.91(0.17) | 0.94(0.17) | 0.76(0.18) |
| | 1000 | -2.04(0.18) | 0.53(0.13) | 0.42(0.13) | 0.62(0.12) | 1.18(0.19) | 1.22(0.19) | 0.41(0.20) |
| | 2000 | -2.21(0.20) | 0.48(0.14) | 0.38(0.14) | 0.59(0.16) | 1.06(0.22) | 1.08(0.21) | 0.81(0.17) |
| 10 | 200 | -0.03(0.16) | 0.28(0.09) | 0.13(0.07) | 0.14(0.08) | 0.10(0.07) | 0.12(0.07) | -0.08(0.14) |
| | 500 | 0.01(0.21) | 0.43(0.13) | 0.31(0.10) | 0.29(0.11) | 0.47(0.12) | 0.47(0.12) | -0.01(0.17) |
| | 1000 | -0.50(0.22) | 0.37(0.14) | 0.30(0.12) | 0.43(0.10) | 0.54(0.12) | 0.53(0.12) | -0.09(0.21) |
| | 2000 | -0.67(0.20) | 0.42(0.13) | 0.33(0.12) | 0.51(0.11) | 0.59(0.16) | 0.60(0.16) | 0.10(0.14) |
| 20 | 200 | 1.46(0.14) | 0.14(0.08) | -0.02(0.06) | -0.01(0.06) | -0.02(0.03) | -0.04(0.06) | 0.17(0.15) |
| | 500 | 1.09(0.15) | 0.25(0.10) | 0.11(0.07) | -0.01(0.07) | -0.07(0.06) | -0.03(0.06) | -0.18(0.16) |
| | 1000 | 0.92(0.18) | 0.48(0.11) | 0.36(0.10) | 0.00(0.11) | -0.10(0.08) | -0.02(0.08) | 0.22(0.18) |
| | 2000 | 0.73(0.22) | 0.24(0.15) | 0.24(0.12) | 0.06(0.11) | 0.18(0.11) | 0.14(0.11) | 0.15(0.19) |
| 50 | 500 | 2.47(0.14) | 0.08(0.09) | -0.02(0.07) | 0.02(0.05) | -0.01(0.03) | -0.08(0.11) | 0.06(0.19) |
| | 1000 | 2.32(0.17) | 0.08(0.12) | -0.02(0.10) | 0.04(0.06) | -0.03(0.03) | -0.13(0.12) | -0.22(0.18) |
| | 2000 | 2.12(0.17) | 0.17(0.13) | 0.18(0.10) | -0.01(0.06) | 0.02(0.04) | 0.00(0.11) | -0.08(0.19) |
| Scenario 2 (Jump points of the feature data density) | | | | | | | | |
| 5 | 200 | -0.72(0.17) | 0.34(0.05) | 0.33(0.04) | 0.51(0.06) | 0.60(0.07) | 0.70(0.07) | 0.32(0.10) |
| | 500 | -1.46(0.15) | 0.42(0.05) | 0.31(0.05) | 0.44(0.06) | 0.92(0.09) | 1.03(0.09) | 0.59(0.11) |
| | 1000 | -1.94(0.13) | 0.48(0.06) | 0.21(0.05) | 0.33(0.07) | 0.99(0.10) | 1.11(0.10) | 0.92(0.11) |
| | 2000 | -1.87(0.17) | 0.46(0.05) | 0.26(0.05) | 0.33(0.06) | 1.43(0.11) | 1.53(0.11) | 1.10(0.11) |
| 10 | 200 | 0.59(0.12) | 0.08(0.05) | 0.03(0.04) | 0.17(0.04) | 0.09(0.03) | 0.13(0.03) | 0.14(0.09) |
| | 500 | 0.44(0.14) | 0.18(0.04) | 0.08(0.04) | 0.05(0.04) | 0.09(0.04) | 0.15(0.04) | -0.07(0.08) |
| | 1000 | 0.27(0.11) | 0.18(0.05) | 0.11(0.04) | 0.18(0.04) | 0.29(0.05) | 0.38(0.05) | -0.11(0.07) |
| | 2000 | 0.02(0.13) | 0.23(0.04) | 0.11(0.04) | 0.17(0.04) | 0.43(0.05) | 0.49(0.05) | -0.12(0.07) |
| 20 | 200 | 1.92(0.12) | 0.08(0.04) | 0.03(0.03) | 0.02(0.02) | -0.01(0.01) | -0.04(0.03) | 0.04(0.07) |
| | 500 | 1.77(0.10) | 0.14(0.05) | 0.01(0.03) | -0.02(0.03) | -0.01(0.04) | -0.02(0.02) | -0.07(0.07) |
| | 1000 | 1.68(0.13) | 0.08(0.05) | 0.02(0.03) | -0.05(0.03) | -0.04(0.02) | -0.03(0.03) | -0.09(0.06) |
| | 2000 | 1.50(0.12) | 0.11(0.05) | 0.06(0.03) | 0.08(0.03) | 0.02(0.02) | 0.09(0.03) | -0.11(0.07) |
| 50 | 500 | 2.85(0.09) | 0.16(0.06) | 0.05(0.04) | -0.01(0.04) | 0.09(0.03) | 0.14(0.06) | -0.04(0.08) |
| | 1000 | 2.90(0.09) | 0.20(0.05) | 0.08(0.04) | -0.03(0.02) | 0.03(0.02) | 0.19(0.06) | -0.10(0.07) |
| | 2000 | 2.82(0.10) | 0.15(0.04) | 0.08(0.03) | -0.01(0.01) | -0.01(0.01) | 0.10(0.04) | -0.12(0.07) |

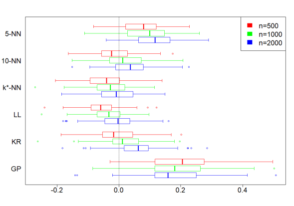
# Experiments on Real Data

- We apply the crystallization learning to several real datasets from the UCI repository.
  1. The CASP dataset (Betancourt and Skolnick, 2001);
  2. The Concrete dataset (Yeh, 1998);
  3. The Parkinson's telemonitoring dataset (Tsanas et al., 2010) for the motor and total UPDRS scores.
- For each dataset, we take 100 bootstrap samples without replacement of size $n$ ($n = 200$, 500, 1000 or 2000) for training and 100 bootstrap samples of size 100 for testing.
- Based on the testing set, we quantify the performance of the method $\mathcal{M}$ by the mean predictive squared error (MPSE),

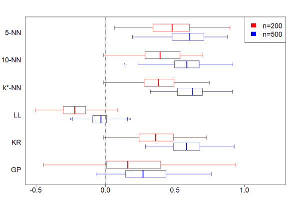
$$\mathrm{MPSE}_{\mathcal{M}} = \frac{1}{100}\sum_{k=1}^{100}\{\hat{\mu}_{\mathcal{M}}(\mathbf{z}_k) - y_k\}^2,$$

where $y_k$'s are responses corresponding to $\mathbf{z}_k$'s.

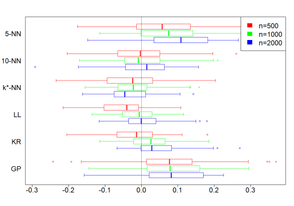
# Experiments on Real Data



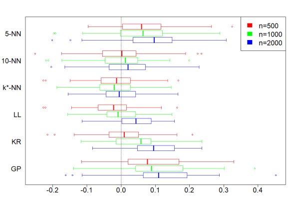

**(a)** CASP dataset




**(b)** Concrete dataset




**(c)** Parkinson's motor UPDRS

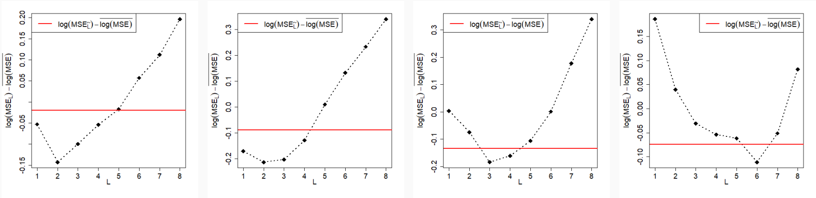



**(d)** Parkinson's total UPDRS

**Figure 8:** Boxplots of $\log(\mathrm{MPSE}_{\mathcal{M}}/\mathrm{MPSE}_{\mathrm{CL}})$ corresponding to existing methods in estimating $\mu(\cdot)$ under different datasets and sizes of the training set $(n)$.

# Experiments on $L$ selection

We conduct experiments to validate the proposed procedure of $L$ selection, which suggests the effectiveness of our LOO-CV procedure in improving the estimation accuracy.




**(a)** $n = 200$ **(b)** $n = 500$ **(c)** $n = 1000$ **(d)** $n = 2000$

**Figure 9:** Averaged values of $\log(\text{MSE}_L) - \overline{\log(\text{MSE})}$ ($L = 1, \ldots, 8$) and $\log(\text{MSE}_{\widetilde{L}}) - \overline{\log(\text{MSE})}$ under different sample sizes ($n$), where $\text{MSE}_L$ is the MSE using the hyperparameter $L$ and $\overline{\log(\text{MSE})} = \sum_{L=1}^{8} \log(\text{MSE}_L)/8$.

# References

Betancourt, M. R. and Skolnick, J. (2001). Universal similarity measure for comparing protein structures. *Biopolymers*, 59(5):305–309.

Chang, T. H., Watson, L. T., Lux, T. C. H., Butt, A. R., Cameron, K. W., and Hong, Y. (2020). Algorithm 1012: DELAUNAYSPARSE: Interpolation via a sparse subset of the Delaunay triangulation in medium to high dimensions. *ACM Transactions on Mathematical Software*, 46(4):1–20.

de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2008). Delaunay triangulations. In *Computational Geometry*, pages 191–218. Springer Berlin Heidelberg.

Hein, M. (2009). Robust nonparametric regression with metric-space valued output. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

Tsanas, A., Little, M., McSharry, P., and Ramig, L. (2010). Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests. *IEEE Transactions on Biomedical Engineering*, 57(4):884–893.

Yeh, I.-C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808.

# End

Thank you for listening.